Credit_Visualization

November 12, 2022

1 Consider the credit card dataset which contains the following columns:

- CLIENTNUM: Primary key of the dataset
- Attrition_Flag: Indicates if a customer is retained or attrited
- Customer_Age: Age of the customer
- Gender: Gender of the customer
- Dependent_count: Number of people dependent on the customer
- Education_Level: Highest level of education of the customer
- Income_Category: Range of income of the customer
- Credit_Limit: Credit card limit
- Total_Revolving_Bal: Pending balance of the credit
- Avg_Purchase: Amount of purchase made by the customer on credit card
- Total_Trans_Amt: Total transaction amount

[42]:
```
#Importing the necessary Libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm
```

[3]:
```
#Importing the required dataset

credit_df = pd.read_csv("CreditCard_DV.csv")
credit_df  #showing a single row in the df
```

[3]:

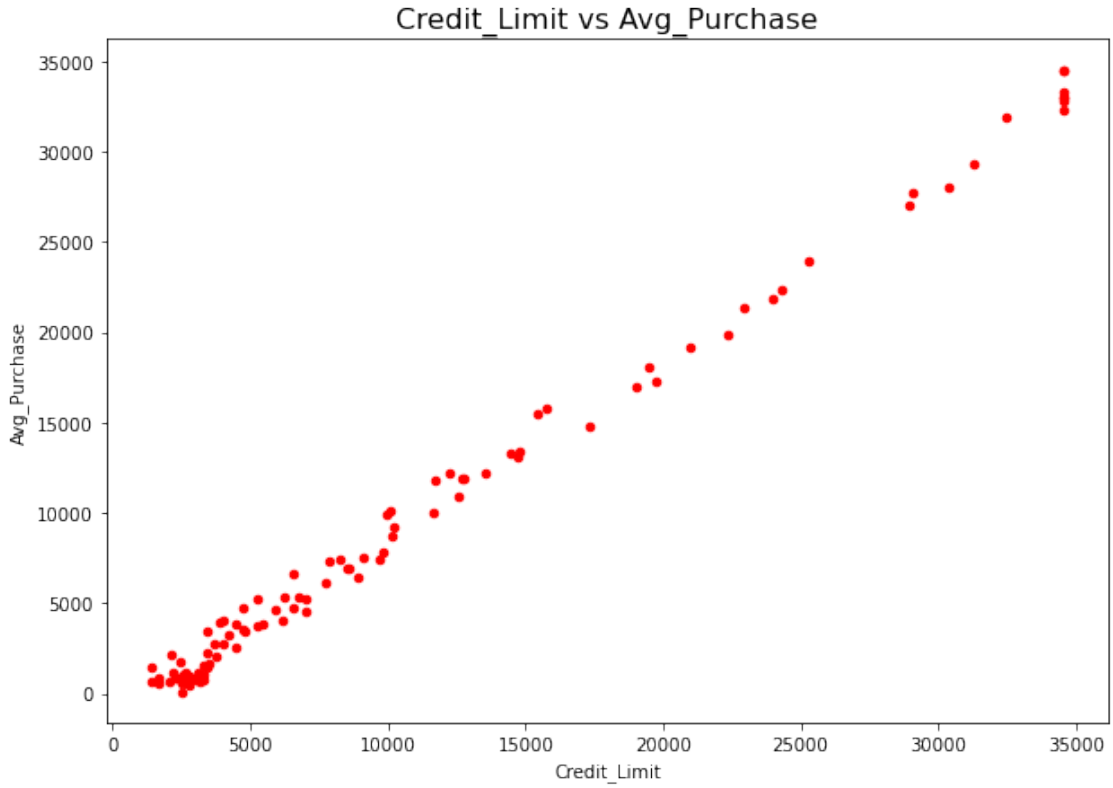
	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	\
0	768805383	Existing Customer	45	M	3	
1	818770008	Existing Customer	49	F	5	
2	713982108	Existing Customer	51	M	3	
3	769911858	Existing Customer	40	F	4	
4	709106358	Existing Customer	40	M	3	
..	...	...	...	...	...	
95	719712633	Existing Customer	64	M	1	
96	772629333	Existing Customer	45	M	3	
97	720336708	Existing Customer	53	M	3	
98	802013583	Existing Customer	56	M	3	

```
99  711887583  Attrited Customer            47      M                2

   Education_Level Income_Category  Credit_Limit  Total_Revolving_Bal  \
0      High School     $60K - $80K       12691.0                  777
1         Graduate  Less than $40K        8256.0                  864
2         Graduate    $80K - $120K        3418.0                    0
3      High School  Less than $40K        3313.0                 2517
4       Uneducated     $60K - $80K        4716.0                    0
..             ...             ...           ...                  ...
95        Graduate  Less than $40K        1709.0                  895
96        Graduate     $40K - $60K        3454.0                 1200
97       Doctorate     $40K - $60K        3789.0                 1706
98         College         $120K +        9689.0                 2250
99         Unknown    $80K - $120K        5449.0                 1628

    Avg_Purchase  Total_Trans_Amt
0        11914.0             1144
1         7392.0             1291
2         3418.0             1887
3          796.0             1171
4         4716.0              816
..           ...              ...
95         814.0             1673
96        2254.0             1313
97        2083.0             1609
98        7439.0             1158
99        3821.0              836

[100 rows x 11 columns]
```

2 Create a bivariate plot to find if there is a correlation between credit card limit and average purchase made on the card.

[4]:
```
#To Plot the data as a scatter plot

ax = credit_df.plot("Credit_Limit","Avg_Purchase",kind="scatter", color =␣
↪"red",marker = "o",figsize=(10,7))

#To add labels and title to the output

ax.set_xlabel("Credit_Limit")        #sets label for x-axis
ax.set_ylabel("Avg_Purchase")        #sets label for y-axis
ax.set_title("Credit_Limit vs Avg_Purchase",fontsize=16)        #sets title for␣
↪the graph
```

[4]:
```
Text(0.5, 1.0, 'Credit_Limit vs Avg_Purchase')
```



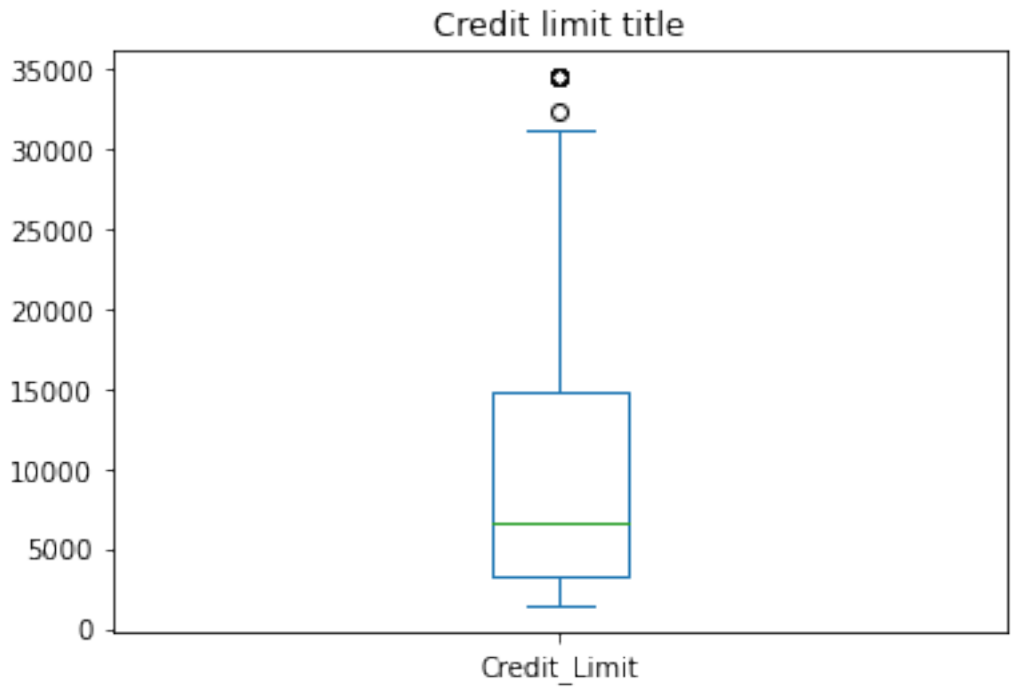
3 Visualise the distribution of values for credit card limit and average purchase made on the card. Also, identify the outliers in the data, if any.

[5]:
```
credit_df["Credit_Limit"].describe()
```

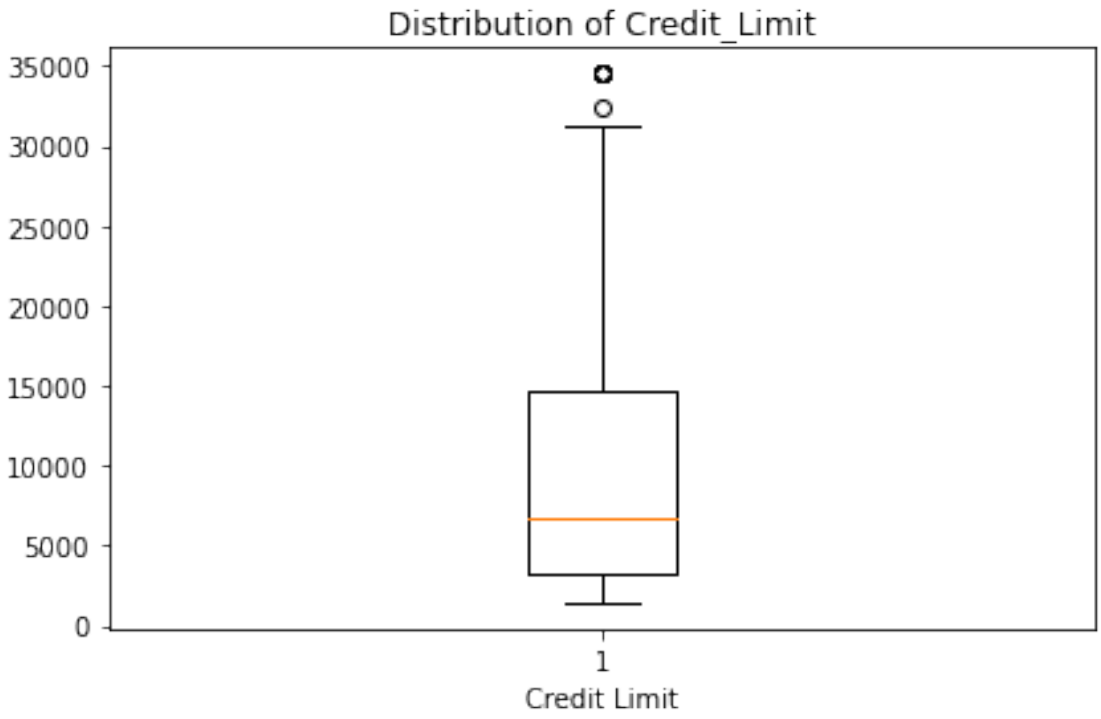
[5]:
```
count      100.000000
mean     10881.756000
std      10056.333148
min       1438.300000
25%       3309.250000
50%       6666.000000
75%      14746.500000
max      34516.000000
Name: Credit_Limit, dtype: float64
```

[6]:
```
ax = credit_df["Credit_Limit"].plot(kind="box")
ax.set_title("Credit limit title")
```

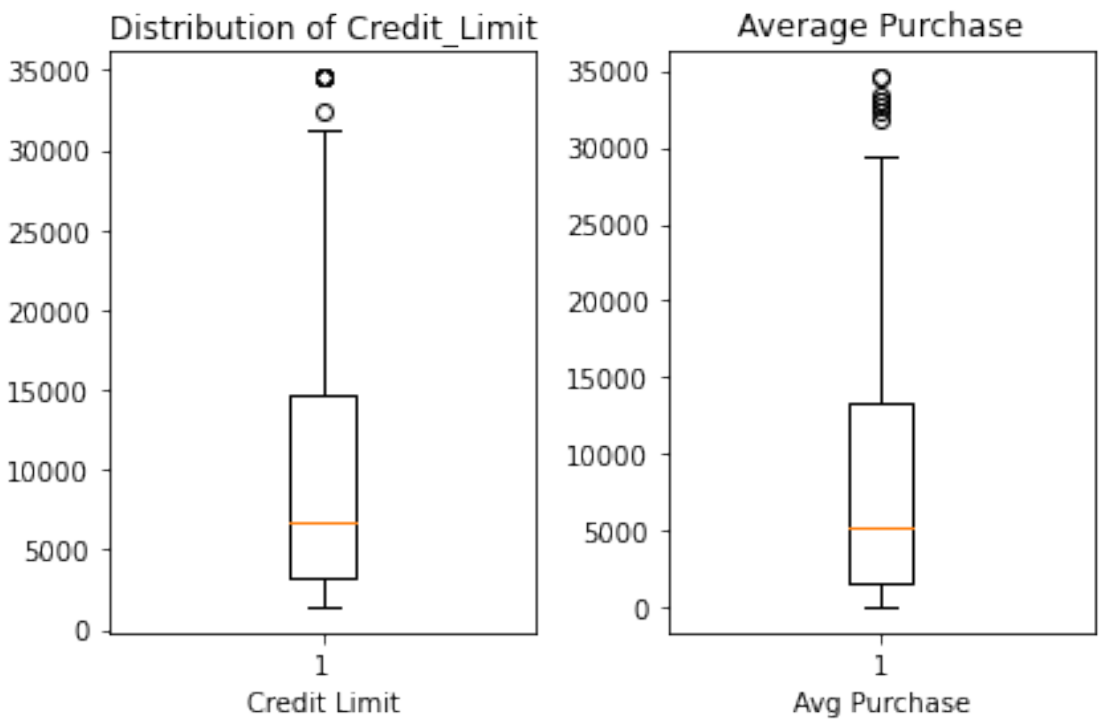
[6]:
```
Text(0.5, 1.0, 'Credit limit title')
```



[7]:
```
fig, ax1 = plt.subplots(1, 1)

#The following lines of code change the alignment from vertical to horizontal
ax1.boxplot(credit_df["Credit_Limit"])

#The following lines of code are used to add labels to axes and title to the␣
 ↪graph

ax1.set_title('Distribution of Credit_Limit')
ax1.set_xlabel('Credit Limit')



#In case of any superimposition of the subplots, the following functions caters␣
 ↪the aesthetics
fig.tight_layout()
```



[8]:
```
fig, (ax1, ax2) = plt.subplots(1, 2)

#The following lines of code change the alignment from vertical to horizontal
ax1.boxplot(credit_df["Credit_Limit"])
ax2.boxplot(credit_df["Avg_Purchase"])

#The following lines of code are used to add labels to axes and title to the␣
 ↪graph

ax1.set_title('Distribution of Credit_Limit')
ax1.set_xlabel('Credit Limit')

ax2.set_title('Average Purchase')
ax2.set_xlabel("Avg Purchase")

#In case of any superimposition of the subplots, the following functions caters␣
 ↪the aesthetics
fig.tight_layout()
```



[9]:
```
cr_limit_arr = credit_df["Credit_Limit"]
# finding the 1st quartile
q1 = np.quantile(cr_limit_arr, 0.25)

# finding the 3rd quartile
q3 = np.quantile(cr_limit_arr, 0.75)
med = np.median(cr_limit_arr)

# finding the iqr region
iqr = q3-q1

# finding upper and lower whiskers
upper_bound = q3+(1.5*iqr)
lower_bound = q1-(1.5*iqr)
print("IQR:",iqr)
print("upper_bound:",upper_bound)
print("lower_bound:",lower_bound)
```

```
IQR: 11437.25
upper_bound: 31902.375
lower_bound: -13846.625
```

[10]:
```
outliers = cr_limit_arr[(cr_limit_arr <= lower_bound) | (cr_limit_arr >=␣
↪upper_bound)]
print('The following are the outliers in the boxplot of Credit Limit:
↪\n',outliers)
```

```
The following are the outliers in the boxplot of Credit Limit:
 6     34516.0
40     32426.0
45     34516.0
61     34516.0
65     34516.0
70     34516.0
81     34516.0
84     34516.0
Name: Credit_Limit, dtype: float64
```

[11]:
```
x = credit_df['Credit_Limit']
v = x[(x == 34516)]
v
```

[11]:
```
6     34516.0
45    34516.0
61    34516.0
65    34516.0
70    34516.0
81    34516.0
84    34516.0
Name: Credit_Limit, dtype: float64
```

[12]:
```
avg_purchase = credit_df["Avg_Purchase"]
# finding the 1st quartile
q1 = np.quantile(avg_purchase, 0.25)

# finding the 3rd quartile
q3 = np.quantile(avg_purchase, 0.75)
med = np.median(avg_purchase)

# finding the iqr region
iqr = q3-q1

# finding upper and lower whiskers
upper_bound = q3+(1.5*iqr)
lower_bound = q1-(1.5*iqr)
print("IQR:",iqr)
print("upper_bound:",upper_bound)
print("lower_bound:",lower_bound)
```

```
IQR: 11790.425
```

```
upper_bound: 31022.887499999997
lower_bound: -16138.812499999996
```

[13]:
```
outliers = avg_purchase[(avg_purchase <= lower_bound) | (avg_purchase >=␣
↪upper_bound)]
print('The following are the outliers in the boxplot of Average Purchase:
↪\n',outliers)
```

```
The following are the outliers in the boxplot of Average Purchase:
 6     32252.0
40    31848.0
45    34516.0
61    34516.0
65    33001.0
70    32753.0
81    32983.0
84    33297.0
Name: Avg_Purchase, dtype: float64
```

4 Provide a visual representation of the number of customers in each income group using a bar chart.

[14]:
```
categories = credit_df["Income_Category"].unique()
categories
```

[14]:
```
array(['$60K - $80K', 'Less than $40K', '$80K - $120K', '$40K - $60K',
       '$120K +', 'Unknown'], dtype=object)
```

[15]:
```
count_df = pd.DataFrame(credit_df[["Income_Category"]].groupby(by=␣
↪"Income_Category").size().reset_index())
count_df.columns = [["Income_Category","Count"]]
count_df
```

[15]:

	Income_Category	Count
0	$120K +	11
1	$40K - $60K	15
2	$60K - $80K	22
3	$80K - $120K	23
4	Less than $40K	22
5	Unknown	7

[16]:
```
count_df.set_index('Income_Category', inplace = True)
count_df
```

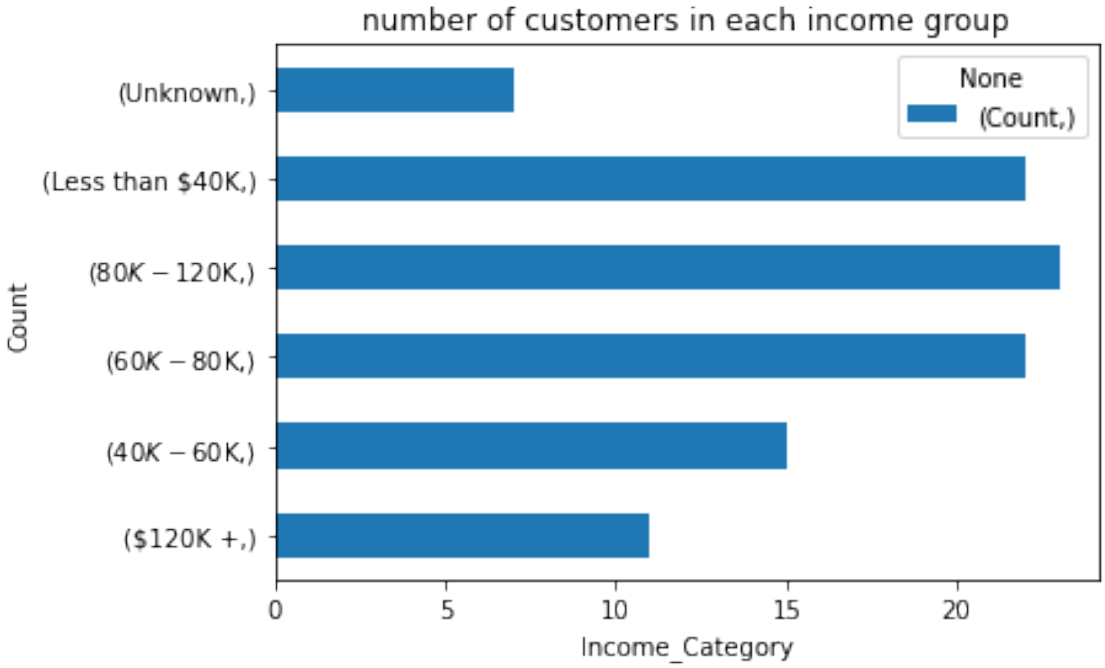
[16]:

Income_Category	Count
($120K +,)	11

```
($40K - $60K,)        15
($60K - $80K,)        22
($80K - $120K,)       23
(Less than $40K,)     22
(Unknown,)             7
```

[17]:
```
count_df['Count'].plot(kind="barh")
plt.title("number of customers in each income group")
plt.xlabel("Income_Category")
plt.ylabel("Count")
```

[17]:
```
Text(0, 0.5, 'Count')
```



5 Plot the frequency distribution of the total transaction amount.

[18]:
```
credit_df["Total_Trans_Amt"].min()
```

[18]:
```
602
```

[19]:
```
credit_df["Total_Trans_Amt"].max()
```

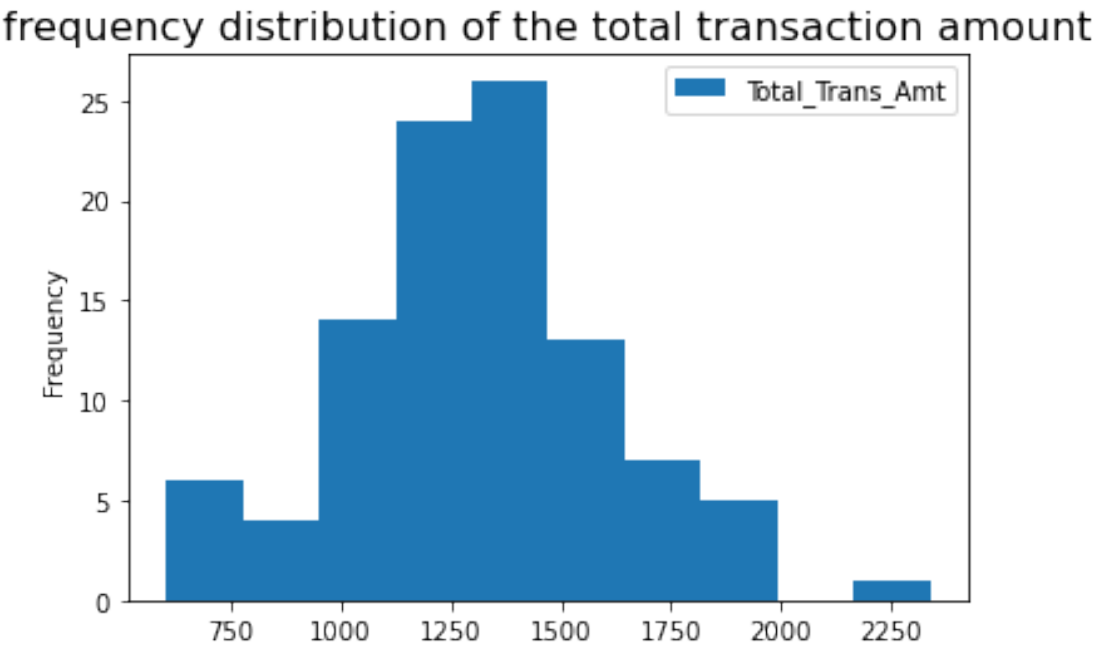
[19]:
```
2339
```

[20]:
```
credit_df["Total_Trans_Amt"].max() - credit_df["Total_Trans_Amt"].min()
```

[20]: 1737

[21]:
```
credit_df["Total_Trans_Amt"].plot(kind="hist")
plt.title("frequency distribution of the total transaction amount", fontsize=16)
plt.legend()
```

[21]: <matplotlib.legend.Legend at 0x26f979d0d30>



6 Graphically represent the percentage of customers retained and those attrited. Highlight the latter by slicing it apart from the main pie

[22]:
```
Attrition_df= pd.DataFrame(credit_df[["Attrition_Flag"]].groupby(by=␣
↪["Attrition_Flag"]).size().reset_index())
Attrition_df.columns = [["Attrition_Flag","Count"]]
Attrition_df
```

[22]:

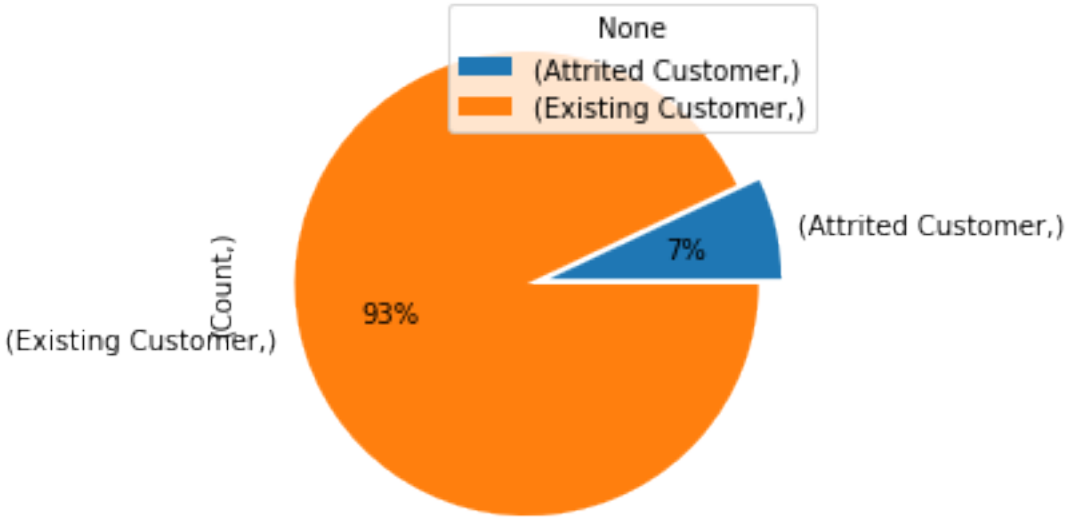
	Attrition_Flag	Count
0	Attrited Customer	7
1	Existing Customer	93

[23]:
```
Attrition_df.set_index('Attrition_Flag', inplace = True)
Attrition_df
```

[23]:
```
                      Count
Attrition_Flag
(Attrited Customer,)      7
(Existing Customer,)     93
```

[24]:
```
explode = (0.05, 0.05)
Attrition_df.plot(kind='pie', y='Count', autopct='%1.0f%%',explode=explode)
```

[24]:
```
<AxesSubplot:ylabel='(Count,)'>
```



7 Consider the Cars93 dataset which contains the following columns:

Manufacturer
Model
Type
Price
MPG.city
MPG.highway Cylinders
EngineSize
Horsepower etc

[25]:
```
#Importing the required dataset

cars_df = pd.read_csv("Cars93.csv")
```

```
columns = ["Manufacturer","Model","Type","Price","MPG.city","MPG.
 ↪highway","Horsepower","Rear.seat.room","Passengers"]
cars_df[columns].head()
```

[25]:

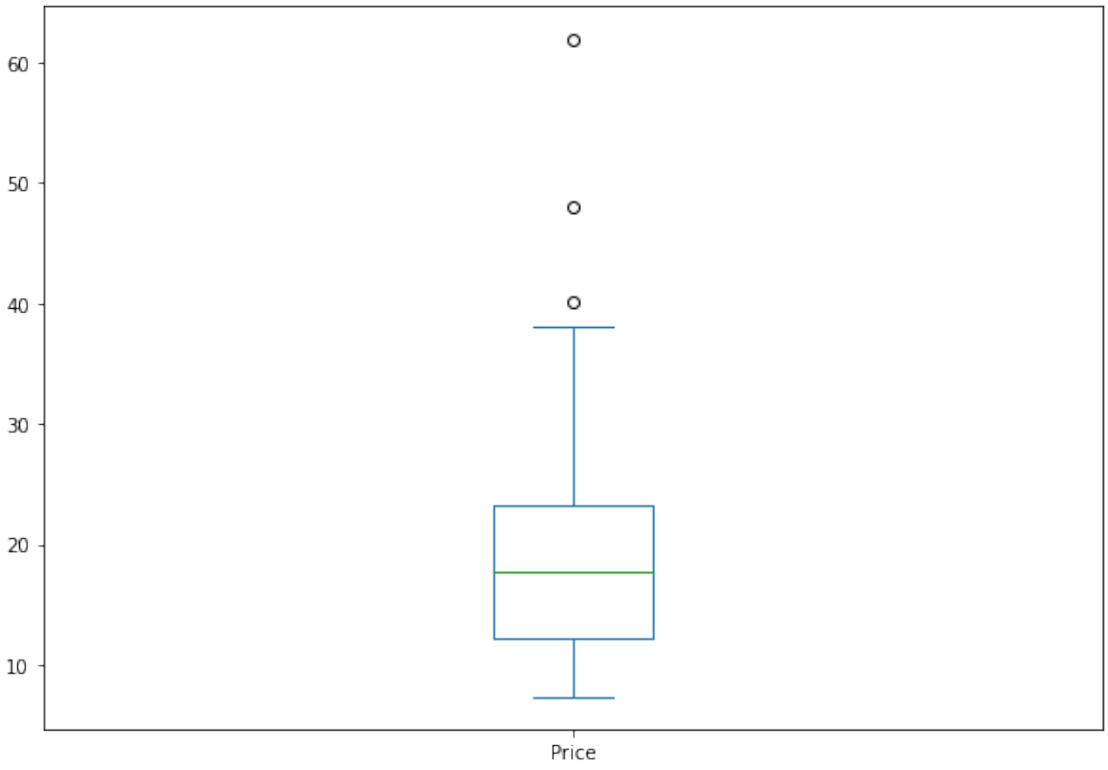
	Manufacturer	Model	Type	Price	MPG.city	MPG.highway	Horsepower	\
0	Acura	Integra	Small	15.9	25	31	140	
1	Acura	Legend	Midsize	33.9	18	25	200	
2	Audi	90	Compact	29.1	20	26	172	
3	Audi	100	Midsize	37.7	19	26	172	
4	BMW	535i	Midsize	30.0	22	30	208	

	Rear.seat.room	Passengers
0	26.5	5
1	30.0	5
2	28.0	5
3	31.0	6
4	27.0	4

8 Visualize the spread of data for the 'Price' column

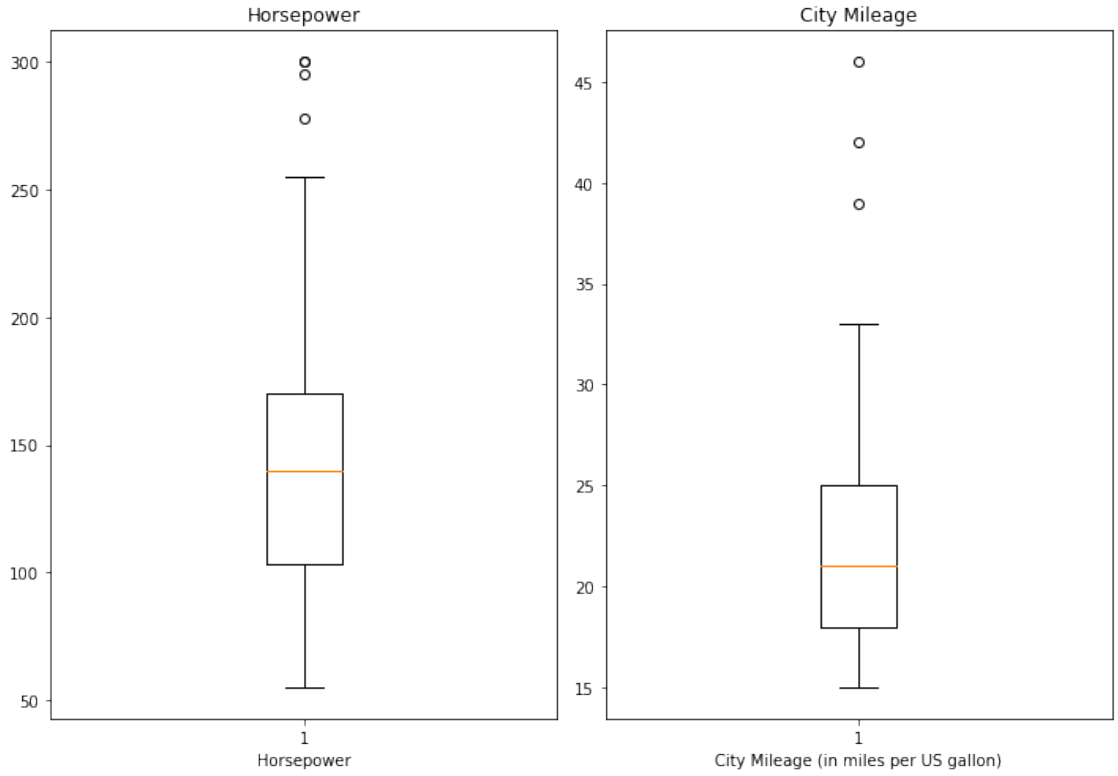
[26]:
```
cars_df["Price"].plot(kind="box",figsize = (10,7))
```

[26]: <AxesSubplot:>



9 Visualize the distribution of price for compact and large type of cars

[27]:
```
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.set_figwidth(10)
fig.set_figheight(7)

#The following lines of code change the alignment from vertical to horizontal

ax1.boxplot(cars_df["Horsepower"])
ax2.boxplot(cars_df["MPG.city"])

#The following lines of code are used to add labels to axes and title to the␣
 ↪graph

ax1.set_title('Horsepower')
ax1.set_xlabel('Horsepower')
ax2.set_title('City Mileage')
ax2.set_xlabel("City Mileage (in miles per US gallon)")
#In case of any superimposition of the subplots, the following functions caters␣
 ↪the aesthetics

fig.tight_layout()
```

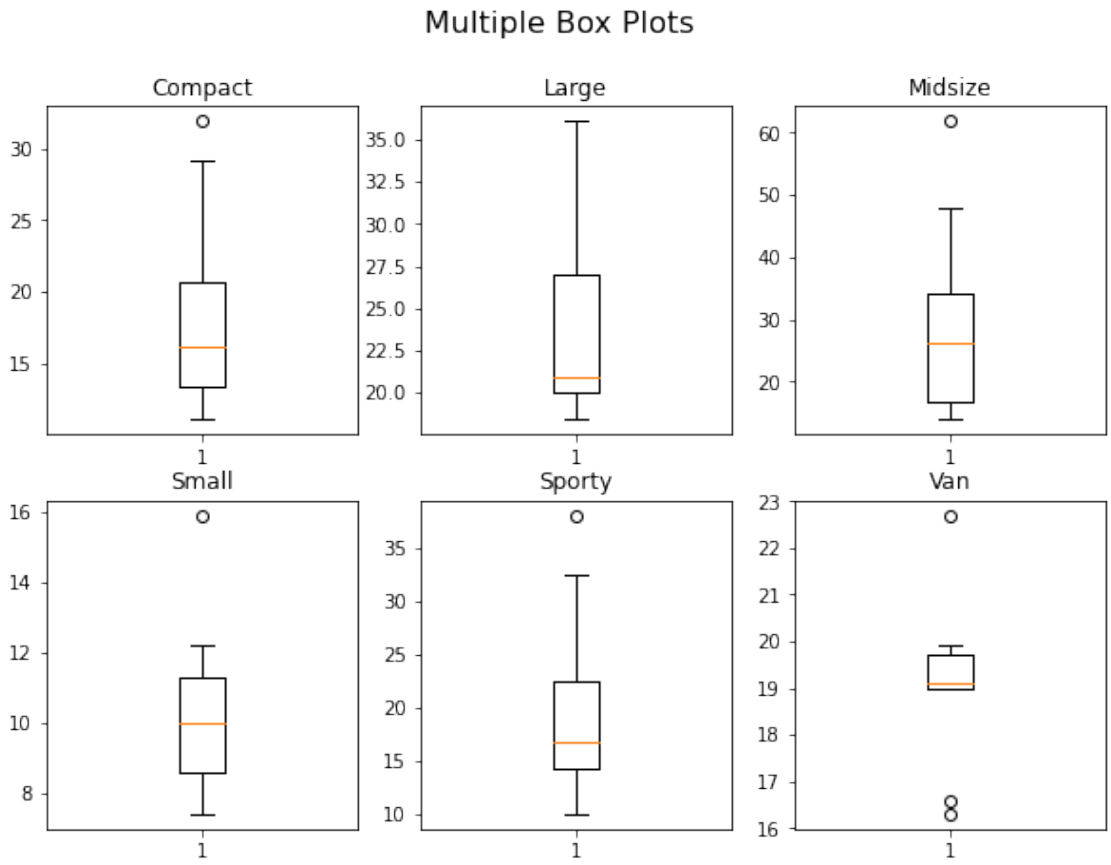


10 Visualize the distribution of price for each type of car

[28]:
```
fig, ax = plt.subplots(2, 3)
fig.set_figwidth(10)
fig.set_figheight(7)
fig.suptitle("Multiple Box Plots", fontsize=16)

ax[0][0].boxplot(cars_df["Price"][cars_df["Type"]=="Compact"])
ax[0][0].set_title('Compact')

ax[0][1].boxplot(cars_df["Price"][cars_df["Type"]=="Large"])
ax[0][1].set_title('Large')

ax[0][2].boxplot(cars_df["Price"][cars_df["Type"]=="Midsize"])
ax[0][2].set_title('Midsize')

ax[1][0].boxplot(cars_df["Price"][cars_df["Type"]=="Small"])
ax[1][0].set_title('Small')

ax[1][1].boxplot(cars_df["Price"][cars_df["Type"]=="Sporty"])
ax[1][1].set_title('Sporty')
```

```
ax[1][2].boxplot(cars_df["Price"][cars_df["Type"]=="Van"])
ax[1][2].set_title('Van')
```

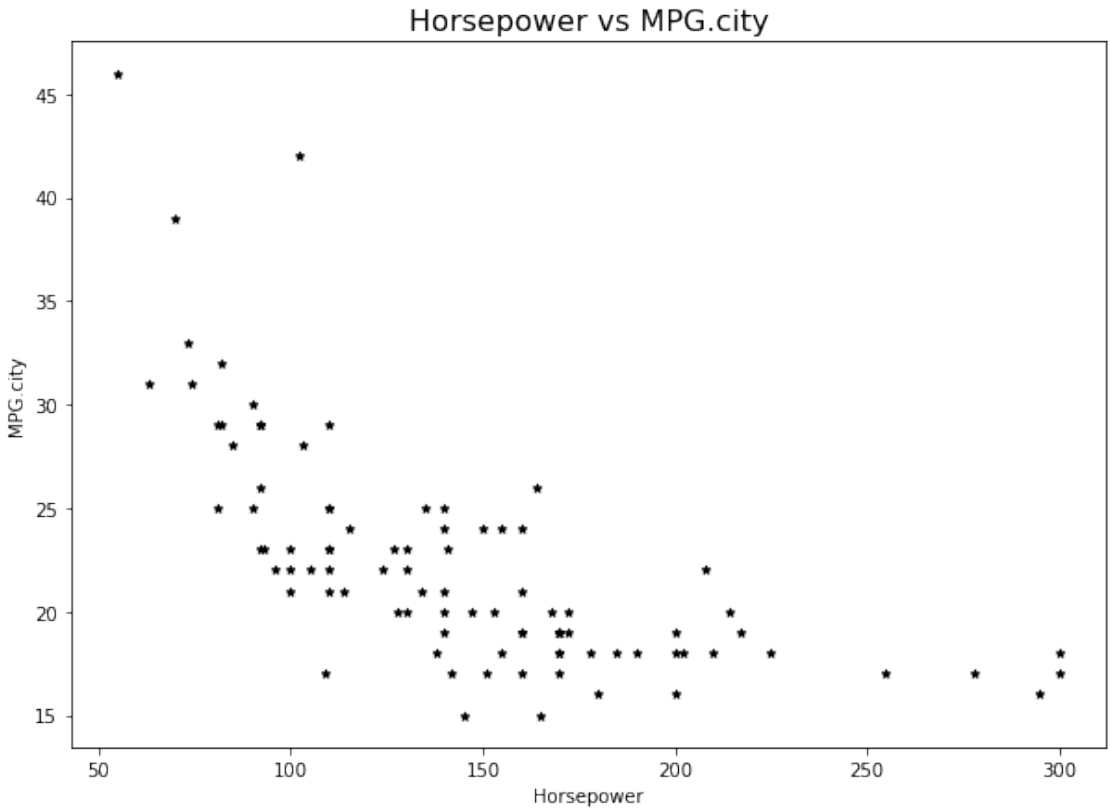
[28]: `Text(0.5, 1.0, 'Van')`



11 Visualize the correlation between Horsepower and Mileage in the city

[29]:
```
ax = cars_df.plot(["Horsepower"],["MPG.city"],kind="scatter", color =␣
↪"black",marker = "*",figsize=(10,7))

#To add labels and title to the output

ax.set_xlabel("Horsepower")        #sets label for x-axis
ax.set_ylabel("MPG.city")         #sets label for y-axis
ax.set_title("Horsepower vs MPG.city",fontsize=16)        #sets title for the␣
↪graph
```

[29]: Text(0.5, 1.0, 'Horsepower vs MPG.city')



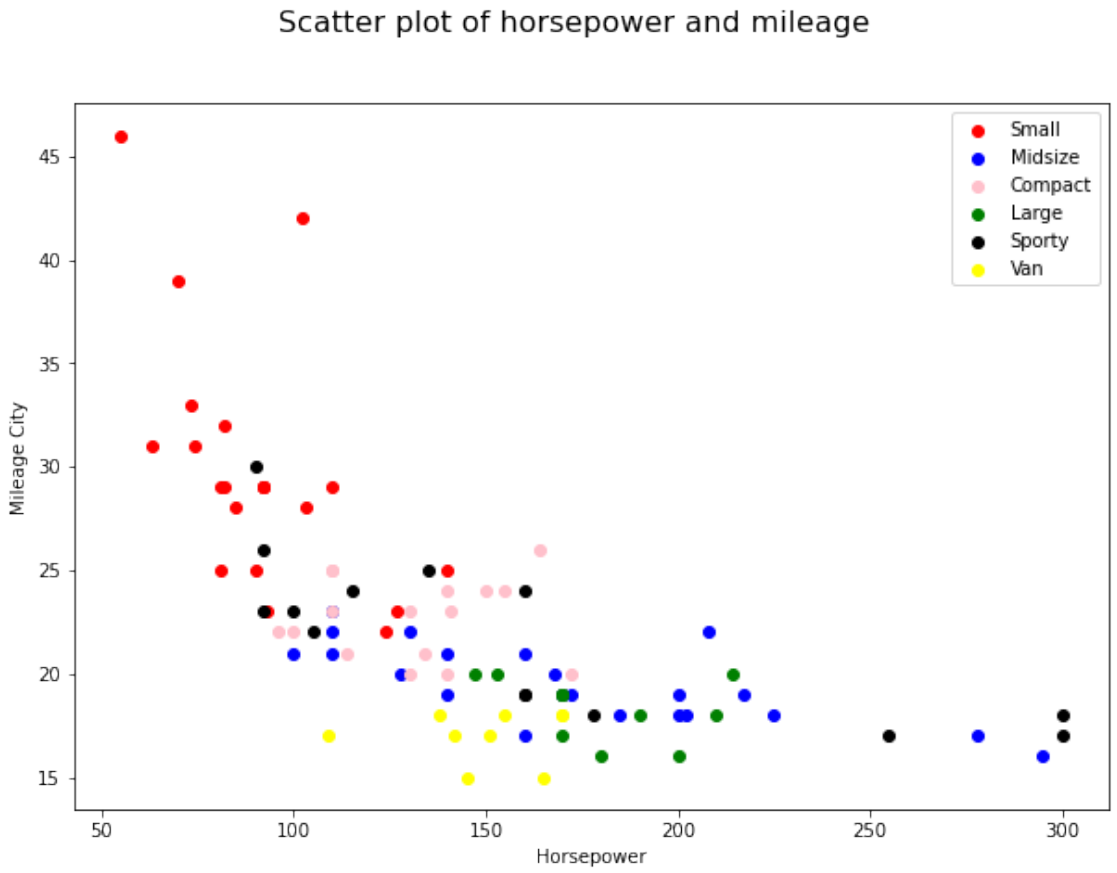
12 Visualize the correlation between Horsepower and Mileage in the city for each type of car

[30]:
```
fig = plt.figure()
fig.set_figwidth(10)
fig.set_figheight(7)

car_type_list = cars_df["Type"].unique()
print(car_type_list)
colors_list = ['red','blue','pink','green','black','yellow']

for car_type,colr in zip(car_type_list,colors_list):         # for every car␣
 ↪type in the car_type_list we plot all the points in the scatter plot
    x = cars_df[cars_df["Type"] == car_type]["Horsepower"]
    y = cars_df[cars_df["Type"] == car_type]["MPG.city"]
    plt.scatter(x,y,color = colr,label=car_type)

plt.suptitle("Scatter plot of horsepower and mileage",fontsize=16)
```

```
plt.xlabel("Horsepower")
plt.ylabel("Mileage City")
plt.legend()
```

```
['Small' 'Midsize' 'Compact' 'Large' 'Sporty' 'Van']
```

[30]: <matplotlib.legend.Legend at 0x26f992770d0>

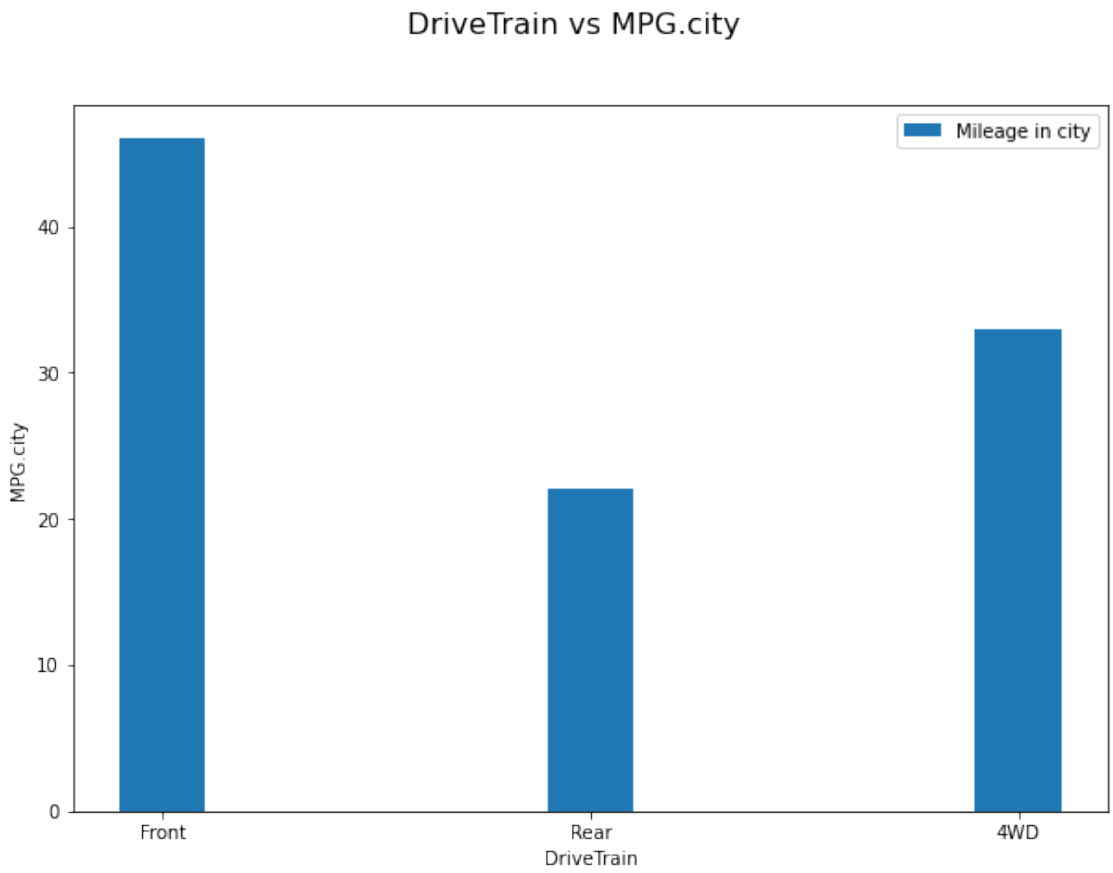


13 Visualize and compare Mileage in the city for each type of DriveTrain using a bar chart

[31]:
```
fig = plt.figure()
fig.set_figwidth(10)
fig.set_figheight(7)
plt.bar(cars_df["DriveTrain"], cars_df["MPG.city"],width=0.2,label="Mileage in␣
↪city")
plt.suptitle("DriveTrain vs MPG.city",fontsize=16)
plt.xlabel("DriveTrain")
plt.ylabel("MPG.city")
```

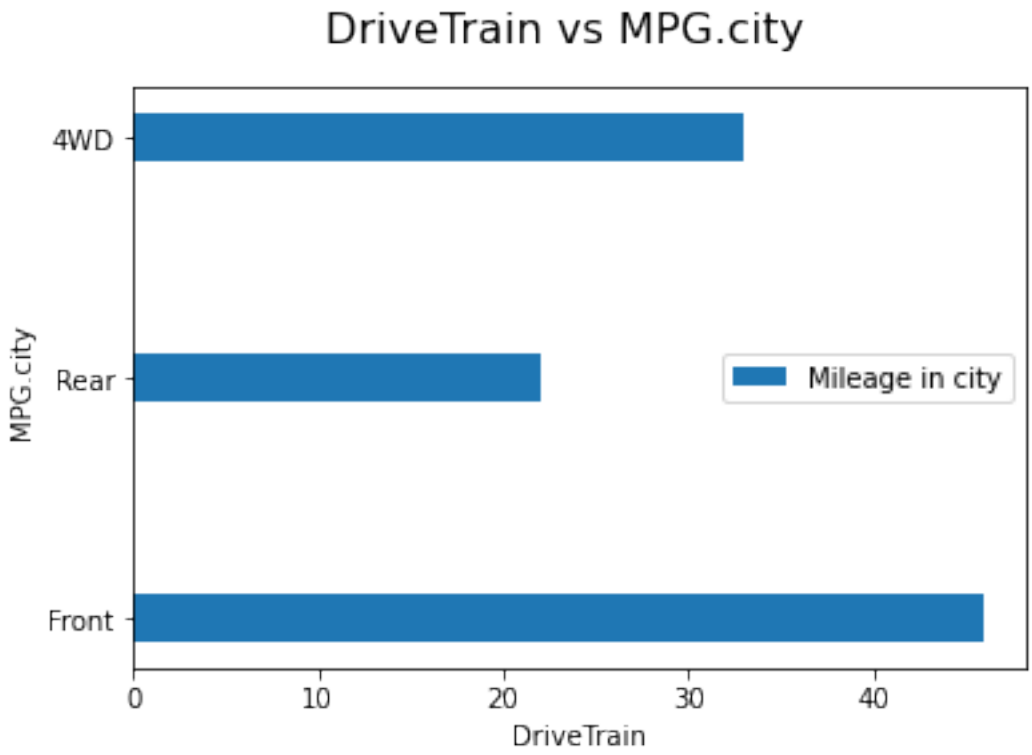
```
plt.legend()
```

[31]: <matplotlib.legend.Legend at 0x26f991f9040>



[33]:
```
plt.barh(cars_df["DriveTrain"], cars_df["MPG.city"],height=0.2,label="Mileage in␣
↪city")
plt.suptitle("DriveTrain vs MPG.city",fontsize=16)
plt.xlabel("DriveTrain")
plt.ylabel("MPG.city")
plt.legend()
```

[33]: <matplotlib.legend.Legend at 0x26f9947f520>



14 Visualize the relationship between "No of Passengers" for each "type of car" using a stacked bar chart

[34]:
```
#Use the following code snippet to filter the unique values of no. of passengers␣
↪a car can carry
cars_df["Passengers"].unique()
```

[34]:
```
array([5, 6, 4, 7, 8, 2], dtype=int64)
```

[35]:
```
#Use the following code snippet to filter the unique values of Types of car.
cars_df["Type"].unique()
```

[35]:
```
array(['Small', 'Midsize', 'Compact', 'Large', 'Sporty', 'Van'],
      dtype=object)
```

[38]:
```
grouped_data = cars_df[["Passengers","Type"]].groupby(by= ["Passengers","Type"]).
↪size()
grouped_data
```

[38]:
```
Passengers  Type
2           Sporty      2
```

```
4           Compact     1
            Midsize     2
            Small       8
            Sporty     12
5           Compact    13
            Midsize    15
            Small      13
6           Compact     2
            Large      11
            Midsize     5
7           Van         8
8           Van         1
dtype: int64
```

[39]:
```
grouped_data = cars_df[["Passengers","Type"]].groupby(by= ["Passengers","Type"]).
 ↪size().unstack()
grouped_data
```

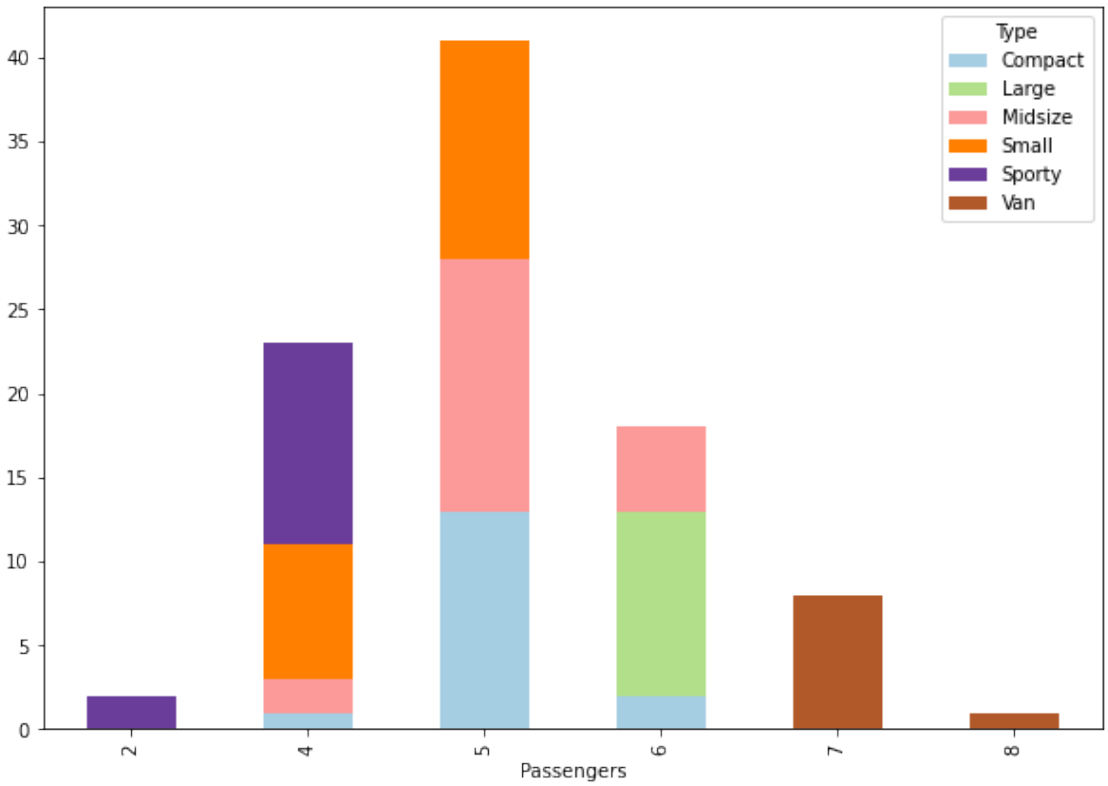
[39]:
```
Type        Compact  Large  Midsize  Small  Sporty  Van
Passengers
2               NaN    NaN      NaN    NaN     2.0  NaN
4               1.0    NaN      2.0    8.0    12.0  NaN
5              13.0    NaN     15.0   13.0     NaN  NaN
6               2.0   11.0      5.0    NaN     NaN  NaN
7               NaN    NaN      NaN    NaN     NaN  8.0
8               NaN    NaN      NaN    NaN     NaN  1.0
```

[40]:
```
#combining the above 2 steps
grouped_data = cars_df[["Passengers","Type"]].groupby(by= ["Passengers","Type"]).
 ↪size().unstack().reset_index()
grouped_data
```

[40]:
```
Type  Passengers  Compact  Large  Midsize  Small  Sporty  Van
0              2      NaN    NaN      NaN    NaN     2.0  NaN
1              4      1.0    NaN      2.0    8.0    12.0  NaN
2              5     13.0    NaN     15.0   13.0     NaN  NaN
3              6      2.0   11.0      5.0    NaN     NaN  NaN
4              7      NaN    NaN      NaN    NaN     NaN  8.0
5              8      NaN    NaN      NaN    NaN     NaN  1.0
```

[43]:
```
#Stacked Bar Graph can be plotted using the grouped data, as follows:
grouped_data.plot(x="Passengers",kind="bar",stacked=True,colormap=cm.
 ↪Paired,figsize=(10,7))
#Matplotlib has built-in colormaps. Here, 'Paired' is used.
```

[43]:
```
<AxesSubplot:xlabel='Passengers'>
```


Type
Compact
Large
Midsize
Small
Sporty
Van
40
35
30
25
20
15
10
5
0
2
4
5
6
7
8
Passengers